PL/I Formac (電子計算機による数式処理)

AUTHOR(S):

鈴木, 久子

CITATION:

鈴木, 久子. PL/I Formac (電子計算機による数式処理). 数理解析研究所講究録 1971, 109: 34-55

ISSUE DATE:

1971-01

URL:

http://hdl.handle.net/2433/106365

数理解析研究所講究録
109巻 1971年 34-55

34

PL/I FORMAC

IBM データセンター 鈴木久子

この記事は，

IBM DATA CENTER User's Guide

SYSTEM/360 PL/I FORMAC

の一部を，講演者の指示に従って，抄録したものである．

PL／I FORMACの言語

PART Ⅱ では，FORMAC LANGUAGEの機能を詳しく説明します。FORMACステートメントを構成するLANGUAGE の要素及びFORMAC Routine についてその機能，使用上の注意，完全なフォーマット（Format)を簡単な例題と共に記述します。

§1 LANGUAGEの要素

FORMACステートメントはFORMAC constant，FORMAC variable，FORMAC function等を用いてFORMAC expressionを作成し，そのFORMAC expression のvalueをFORMAC variableにassignすることであり，PL／Iステートメントとはkey-word "LET" を使用することにより識別されます。

1.1. FORMAC assignment statement

FORMAC assignment statementのFormatは次の様になります。

LET ($v_1 = e_1$; $v_2 = e_2$; …… $v_n = e_n$) ;

ここでv_nは FORMAC variableであり，e_nはFORMAC expression でありまず。このステートメントが実行されると，右辺のexpression e_n を構成しているFORMAC variableのcurrent value を使用して数式処理が行なわれ，その値が左辺のFORMAC variableのcurrent value となります。PL／Iステートメントとの識別はKeyword "LET"によって行なわれ，1つのLETステートメントLET ( ) ；で任意の数のassign-ment statementを実行させることが出来ます。

＜例＞

```
LET (A=B+C) ;
LET (A=B+3 ; C=A+X ; D=SIN (Y)) ;
```

1.2. FORMAC constants

FORMAC constantsには4種類あります。即ち，小数点付実数，整数，有理整数，システム定数であります。

a. Floating point numbers

Floating Point Numberは，1～9桁の小数点つき数字である。又decimal exponentを用いることも出来ます。Floating point numberは約5.4×10^{-79}から7.2×10^{75}までの数を表わすことが出来ます。Floating point numberは長精度（Double Precision）で記憶されており，又その演算は長精度で行なわれます。

＜例＞

```
.0032, 6.0, -1.263E23, .034768817E-5, 4E3
```

exponent を使用するときは小数点を省略してもよい。（4E3 ≡ 4000.）

b. Integer constants

1～2295桁の整数を表わすことが出来ます。IntegerとFloating point numberとの混合演算ではその結果はFloating point numberになる。IntegerとIntegerのoperation（+, −, **, *）はIntegerになります。又除算は一般には有理整数となります。

c. Rational numbers

有理整数はInteger同志の除算によってInteger／Integerの形で表わされる。Rational number同志の演算はRational mode arithmeticを使用して行ない，その結果はいつも簡約化される。特に分子が分母で割り切れるときは，その結果はIntegerとなります。Floating point numberとの混合演算はFloating point numberとなります。

＜例＞

LET(A＝1／6＋1／3; B＝7.3*A; C＝8*6／12; D＝8*7／12);

Result：A← 1／2, B← 3.65, C← 4, D← 14／3

d. System constants

前もってFORMACが備えている3個の定数があります。

#E（$e \equiv 2.7812\cdots\cdots$）　自然対数のbaseを表わします。

#P（$\pi \equiv 3.14159\cdots\cdots$）　円周率を示します。

#I（$i \equiv \sqrt{-1}$）　虚数単位を示します。

これらのSystem constantは使用方法によって，自動的に変換が行なわれます。

＜例＞

LET(A＝#I**2; B＝#E**LOG(X); C＝SIN(#P／6));

Result：A←−1, B←X, C← 1／2

〔注意〕FORMAC systemがIntegerを必要とする場合（例えば，subscripts，FAC及びCOMBのArgument等）Rational number又はFloating point numberはいつも4捨5入してIntegerになります。

＜例＞

3.4→3, 3.5→4, 7／3→2, 3／2→2, −3.4→−3, −3.5→−4

1.3. FORMAC variable

一般にFORMACステートメントに現われるvariableはFORMAC variableと見られます。FORMAC variableの名前のつけ方は次に記した点を除いてPL／I variableの名前のつけ方と同じであります。

① FORMAC variableの名前の長さは最高8文字で，underscore（−）を含んではなりません。又DEN×××の形はFORMAC routineで使用されているので，さけなければなり

ません。

② FORMACがリザーブ（reserve）している記号（例えば，SIN, LOG, EVAL等）の使用はさけること。

③ FORMAC variableはbase, scale, precisionを持っていないのでDECLAREする必要はありません。

次にFORMAC variableの便利な事柄を記述します。

① たとえFORMAC variableとPL/I variableで同じ名前を使用しても相異なるものと見なされます。

<例>

A＝5；LET（A＝6）；B＝A；　　Result：B←5

② FORMAC variableは4元まで使用可能であります。サブスクリプトはFORMAC Integerか又は計算結果がIntegerになるFORMAC expressionであり，そのサブスクリプトの範囲は−31622～＋31622まで許されます。又サブスクリプトのついたFORMAC variableはDECLAREする必要もなければDIMENSIONの定義もする必要がありません。

機能上FORMAC variableは，Atomic variable及びAssigned variableの2種類に分けることが出来ます。

Atomic variableはvalue（数値又は数式）を持たないvariableで変数自身を表わしています。従ってFORMACステートメントの左辺には現われません。Assigned variableはvalueを持つvariableで少なくとも一度はFORMACステートメントの左辺に現われます。

1.4. FORMAC expression

FORMAC expressionは本質的にはPL/I expressionと同じ様に書かれます。FORMAC variablesとconstant（＋，−，＊，／，＊＊）で結ばれ，その計算順序はPL/I expressionの時と同様，普通の計算法則が適用されます。FORMAC expressionが実行されると，右辺のexpressionに現われるassigned variableはその時点での値（current value）を使用して計算します。然しながら，次の例のような場合は注意を要します。

<例>

① LET（A＝B＋C；B＝X＊Y；E＝A＋B）；

② LET（B＝X＊Y；A＝B＋C；E＝A＋B）；

①のステートメントが実行されるとA←B＋C，B←XY，E←B＋C＋XYとなり，A＝B＋Cの実行時にはBはatomic variableと考えられます。B＝X＊Y；の実行でBはAssigned variableと見なされXYの値をもち，E＝A＋B；の実行においてA＝B＋C，B＝XYのCurrent valueを代入します。A＝B＋CのBの変数にB＝XYの値を代入してからEの計算を行なうというようなことはしません。即ち“Back-Substitution”は行なわれません。一方②のステートメントの実行ではB←XY，A←XY＋C，E←2XY＋Cとなります。

1.5. FORMAC function

FORMAC functionとしてはFORMAC expressionをArgumentとしてとる4種類のfunctionがあり，これらのfunctionはPL/I-Like function, Integer-valued function, user-defined function, function variableがあります。

a. PL/I-like function

PL/Iにおいてnumeric argumentをとる次の様なfunctionはFORMACにおいてはsymbolic argumentと共にFORMAC expressionの中で使用することが出来ます。

SIN　COS　SINH　COSH　ATAN　ATANH　ATAN(x, y)

LOG　LOG10　LOG2　ERF

〔注意〕

これらのfunctionはOPTSET(TRANS);が指定されているときは，これらのfunctionのargumentが数値(floating point, integer, rational)であるときは自動的にfunctionの計算が行なわれます。又OPTSET(NOTRANS);が指定されているときには，argumentはsymbolicのまま残っていて計算は行なわれません。

<例>

LET(A=LOG(X**2)+SIN(3/4));

Result: A←LOG(X^2) + .68163876　(OPTSET(TRANS)の時)

A←LOG(X^2) + SIN(3/4)　(OPTSET(NOTRANS)の時)

又次の様なfunctionがFORMAC expressionの中に使用されると変換されます。

function	translation
EXP(X)	#E**X
SQRT(X)	X**(1/2)
SIND(X)	SIN(1/180*#P*X)
COSD(X)	COS(1/180*#P*X)
TAN(X)	SIN(X)/COS(X)
TAND(X)	SIN(1/180*#P*X)/COS(1/180*#P*X)
TANH(X)	SINH(X)/COSH(X)
ATAND(X)	180*ATAN(X)/#P
ATAND(X,Y)	180*ATAN(X,Y)/#P
ERFC(X)	1-ERF(X)

<例>

LET(A=EXP(SIND(X**2)));　Result: A← #E $^{SIN(1/180\#PX^2)}$

b. FORMAC integer-valued function

FORMAC expressionの中で使用することの出来るものでIntegerの値をもつ

functionとしてFAC, COMB, STEPの3種類があります。

FAC $(n) = n!$

COMB $(n, n_1, n_2, \cdots\cdots, n_k) = \dfrac{n(n-1)\cdots\cdots(n-n_1-\cdots\cdots-n_k+1)}{n_1!\, n_2! \cdots\cdots n_k!}$ $\left(\sum_{i=1}^{k} n_i \leq n\right)$

〔注意〕

OPTSET controlとしてOPTSET(INT)が指定されるとIntegerに計算される。

OPTSET(NOINT)の時はsymbolicとしてそのまま残っています。

<例>

LET(X=FAC(5); Y=COMB(5;2));

Result: X← 120, Y← 10(INT指定のとき); X← 5!, Y←(5; 2)(NOINTのとき)

次にSTEP functionはSTEP(e_1, e_2, e_3)の形をとり, expression e_1, e_2, e_3が数として評価されるなら次の様に定義されます。

STEP$(e_1, e_2, e_3) = 1$ $e_1 < e_2 < e_3$ 又は$e_1 = e_2 = e_3$の時

$= 0$ Otherwise

但し, e_1, e_2, e_3 が数値でないとき, あるいはOPTSET(NOINT)の指定があるときは, Symbolicとして残ります。

<例>

LET(Y=STEP(−1, X, 0)+SEP(0, X, 1) * COS(X));

Result: Y← 0 X<−1, X>1の時

Y← 1 −1<X<0の時

Y← COS(X) 0<X<1の時

c. Function Variables

FORMACでは任意の関数を数式を定義することなくfunction Variableとして使用することが出来ます。

一般の形 function name.$(e_1, e_2, \cdots\cdots, e_n)$

<例>

LET(Y=F.(A+B, A−B)+SIN(G.(X+3)));

ここでF.(A+B, A−B)はA+B及びA−Bを変数とする関数を表わし, G.(X+3)はX+3を変数とする関数を示しています。又function variableはsubscriptをつけることも出来ます。Y(I, J).(A, B, C) ここでI, Jはsubscriptであります。更にfunction variableは他のfunction variableにvalueとしてassignすることも出来ます。

<例>

LET(F=G; X=F.(A, B));

Results: X← G.(A, B) となります。

d. User-defined functions

FORMAC では数式を明確に定義する function も使用することが出来ます。

FNC(function name)=expression($(1), $(2), …, $(n))

ここで$(1), $(2), …, $(n) はexpressionに従って, function の変数を示しています。

<例>

LET(FNC(F)=3*COS($(1)**2+$(2))+1);

LET(X=B+C; Y=F(X+4, 3)+2);

この例で function F≡F($(1), $(2))=3*COS($(1)**2+$(2))+1 として定義されており, Y←3 COS((B+C+4)²+3)+3 となります。

〔注意〕

function は使用される前に定義されなければならない。又 function の定義の中で使用されたFORMAC variableは, function の定義された時点におけるFORMAC variable の current valueが使用されます。更に前似って定義した function をFNCステートメントの中で使用することも出来ます。

<例>

```
LET(B=5;
        FNC(R)=B*COS($(1)+2*$(2));
        FNC(S)=R($(1), B)**3;
          ⋮
        B=6;
        Y=R(A+B, B*X);
        Z=S(ALPHA));
```

この結果はFNC(R)ではR≡R($(1), $(2))=5*COS($(1)+2*$(2)) と定義されFNC(S) ではS≡S($(1))={5*COS($(1)+2×5)}**3と定義され, Yの計算ではB=6が使用されます。従ってY←5 COS(A+12X+6), Z←125COS³(ALPHA+10) となります。

1.6. CHAIN

CHAIN operationによって, FORMAC expressionのlist(又はChain) を作りFORMAC variableにassignすることが出来る。これはFORMAC functionやroutine の argument を作成するのに使います。

<例>

LET(X=A+3*SIN(B); Y=CHAIN(A, X+A, B+7, 4));

Results: Y←(A, 2A+3SIN(B), B+7, 4)

LET(X=(A, B+2, C); Y=F.(X, D));

Results : Y←F (A, B+2, C, D)

なお, chain を定義するときには, "CHAIN" を省略してもよい。X=(A, B+2, C); は X=CHAIN(A, B+2, C); と同じことを示します。

1.7. FORMAC output

FORMAC output としてPRINT_OUTステートメントがあります。これはFORMAC variable の valueをプリントアウトします。

<例>

LET(X=A** 5+SIN(3*A) ** 2+2/3) ; PRINT_OUT(X) ;

Results : $X=A^5+SIN^2(3A)+2/3$ とプリントアウトされます。

〔注意〕

① 1個のPRINT_OUTステートメントで複数のFORMAC variableをプリントすることが出来ます。即ちPRINT_OUT(X;Y;Z); と書くと, 各ラインにそれぞれ変数とその value がプリントアウトされます。

② atomic variable XをPRINT_OUT(X) とすると, X= とプリントされます。

③ Floating Point number は9桁の文字と小数点でプリントされます。10^{-2} ～ 10^6外の数はE format でプリントされます。

④ Keyword "LET"はPRINT_OUT で置きかえることが出来ます。

<例>

LET(A=B+C); PRINT_OUT(A);

LET(D=SIN(X)); PRINT_OUT(D);

これは次の様に書くことも出来ます。

PRINT_OUT(A=B+C ; D=SIN(X)) ;

§2 FORMAC routine

FORMACにおいてはFORMAC expressionの数式処理を行なうのに各種のroutineが備えられています。然しながらFORMAC routineは, 幾つかの例外を除いてFORMAC expression を valueとして送ってくれる function であります。又FORMAC routine のargumentはその routineがcallされる前に計算されます。

2.1. User-Controlled Simplification

このroutineにはexpressionを展開する為のroutineと, a/b形の expression にする為のroutineとがあります。

a. Expansion Routines : MULT, DIST, EXPAND

Format : (i) MULT(*e*)

(ii) DIST(*e*)

(iii) EXPAND(*e*)

Result :

(i) MULTはexpression eの中に現われる"ベキ乗"を展開する。NOINT optionがsetされていると展開されて出来る係数をcombinatorialを使用して表わします。

<例>

LET(X=(A+B)*(A−B)**2 ; Y=MULT(X)) ; Y←(A+B)(−2AB+A²+B²)

OPTSET(NOINT) ; LET(X=5*(A+B)**3 ; Y=MULT(X)+A**5) ;

Y←5((3 ; 1) B²A+(3 ; 1) BA²+A³+B³)+A⁵

ここで(3 ; 1)はCOMB(3, 1)を意味しています。

(ii) DISTはexpression eに分配法則(distributive Law)を適用します。

即ち $(a+b)(c+d) \rightarrow ac+ad+bc+bd$, $a(b+c) \rightarrow ab+ac$, $(a+b)/c \rightarrow a/c+b/c$

DISTはMULTの働きを含んでいません。

<例>

LET(P=DIST(C*(A+B)) ; Q=DIST((A+B)/C)) ; P←AC+BC, Q←A/C+B/C

LET(Y=(X−A)**2+2*(X−A)**4+3*(X−A)**6 ; Z=DIST(2*(X−A)**3*Y)) ;

Z←2(X−A)⁵+4(X−A)⁷+6(X−A)⁹

(iii) EXPANDはMULTとDISTの機能を両方とも備えていて, expressioneを展開します。

<例>

LET(X=A*(COS(B(C+D))+A) ; Y=3*(A−2)**2 ; Z=EXPAND(X−Y)) ;

Result : Z←12A+COS(BC+BD)A−12A²−12

LET(X=(A**2+A*B*C)/(A*B) ; Y=EXPAND(X)+4*C) ;

Result : Y←A/B+5C

LET(Z1=3+4*#I ; Z2=−2+7*#I ; PRODUCT=EXPAND(Z1*Z2)) ;

Result : PRODUCT←−34+13#I

〔注意〕

MULT, DIST及びEXPANDはfactored denominators(因数化された分母)は完全に展開はしない。

<例>

LET(X=3*(Q+7)/(P*(Q+R)**3) ; Y=EXPAND(X)) ;

Result : Y←3Q/(P(Q+R)³)+21/(P(Q+R)³)

更にOPTSET controlとしてOPTSET(EXPND)が指定されるとすべてのexpressionの計算で展開が自動的に行なわれます。

b. Common Denominator Functions : CODEM, FRACTN

Format (i) CODEM(e)

(ii) FRACTN(e)

Result : 両方のfunction共expression eをa/bの形に変換する。

(i) CODEMはすべてのレベルにおけるexpressionをa/bの形にする。

(ii) FRACTN(e)は外側のレベル（outermost level）だけにおいてa/bの形にする。

<例>

LET(X＝A/(B+C/D)+E/F；Y＝CODEM(X)；Z＝FRACTN(X))；

Result：Y← (FDA+(C+DB)E)/((C+DB)F)

Z← (FA+(B+C/D)E)/((B+C/D)F)

2.2. Substitution routine

FORMACにおいて，substitution routineにはEVALとREPLACEがあります。これらのroutineはargumentのexpressionに任意のexpressionを代入して計算する働きを持っております。

a. EVAL

Format：　　EVAL($e, a_1, b_1, a_2, b_2, \cdots\cdots, a_n, b_n$)

Result：

EVALはargument expression eの中のatomic variable，system constant，又はfunction a_iに任意のexpression b_iを代入します。

① a_iがatomic variable又はSystem constant(#E, #I, #P)の時はexpression eの中にa_iが現われる度にb_iを代入します。

② a_iがfunction即ちf($(1), $(2), …, $(m))，$b_i$が$\varepsilon$($(1), $(2), …, $(m))のときexpression eの中の$f(arg_1, arg_2, \cdots, arg_m)$は$\varepsilon(arg_1, arg_2, \cdots, arg_m)$で代入されます。

③ 代入方法は並行代入（parallel substitution）であります。

<例>

LET(X＝#P*(R−RHO)**2；Y＝EVAL(X, #P, 3.141592, R, A+B))；

Y← $3.141592(A+B-RHO)^2$

LET(X＝CHAIN(A, B, B, E, C, A)；Y＝A*(A+B)+C*(A+D)；

Z＝EVAL(Y, X))；

XはCHAINによってリストX←(A, B, B, E, C, A)になっていますからZ＝EVAL(Y, X)はZ＝EVAL(Y, A, B, B, E, C, A)と同じことを示します。又(A, B), (B, E), (C, A)をpairにして代入されますがAにBを，BにEを代入，従ってAにEを代入するとはなりません。それぞれ独立して代入されます。（parallel substitution）

従って，結果はZ←B(B+E)+A(B+D)となります。

<例>

LET(X＝F.(A−B, A+B)+G.(C**2)**2；

Y＝EVAL(X, F.($), $(1)**2+$(2), G.($), COS($(1))))；

Y ← (A−B)²+(A+B)²+COS²(C²)

〔注〕 F.($(1), $(2)), G($(1))の代りにF.($), G($) を使用することが出来ます。

b. REPLACE

Format : REPLACE(e, a_1, b_1, a_2, b_2, ……, a_n, b_n)

Result :

a_i, b_i は任意のFORMAC expressionであり, expression eの中の subexpression a_iに subexpression b_iを代入します。代入方法は連続代入 (serial substitution) で次の様に行なわれます。即ちexpression eの中のa_1に b_1 を代入し, その結果得られるexpression e_1とするとe_1 の中の subexpression a_2 にsubexpression b_2 を代入します。同様にして (a_n, b_n) まで行なわれます。

<例>

LET(Y=A*X**2+B*X+C*D+E ; Z=REPLACE(Y, X**2, T(2), X, T(1), C*D*E, CONST)) ;

Z ← AT(2)+BT(1)+CONST

<例>

LET(X=CHAIN(A, B, B, E, C, A) ; Y=A*(A+B)+C*(A+D) ; Z=REPLACE(Y, X)) ;

Z ← 2E²+A(E+D)

2.3. Analytic Differentiation Routines : DERIV, DIFF, DRV

FORMAC では3種類の微分関係のroutineがあります。

DERIVは任意の変数で, expressionを偏微分します。DIFFはfunction variable の偏導関数を取り出します。又DRVはfunction variable の argumentに関して微分します。

a. DERIV

Format : DERIV(e, V_1, n_1, V_2, n_2, ……, V_r, n_r)

Result :

eはexpressionで, V_iはatomic variable, n_iは非負の整数であるとき求められる結果は次の様になります。

$$\frac{\partial^n}{\partial V_1^{n_1}\partial V_2^{n_2}\cdots\partial V_r^{n_r}}\cdot e \qquad \left(\text{ここで } n=\sum_{i=1}^{r} n_i\right)$$

なお, n_i が1のときは省略してもよい。

<例>

LET(P=3*SIN(A*X)+A*X**2 ;
Q=DERIV(P, X) ; R=DERIV(P, X, 2, A, 1)) ;

Q ← 3A COS(AX)+2AX, R ← −3A²X COS(AX)−6ASIN(AX)+2

<例>

LET(Q=DERIV(F.(A, B, C), A, 2, C, 1) ; R=DERIV(G.(A), A, 3)) ;

$Q \leftarrow F^{(1^2\,3)}.(A, B, C)$, $R \leftarrow G^{(1^3)}.(A)$

LET(X=DERIV(A**2+F.(A, B), A, 2) ;

Z=DERIV(F.(A+B, 2*A*B), A, 2)) ;

$Y \leftarrow F^{(1^2)}.(A, B)+2$, $Z \leftarrow F^{(1^2)}.(A+B, 2AB)+4BF^{(12)}.(A+B, 2AB)+4B^2F^{(2^2)}.(A+B, 2AB)$;

<例>

LET(S=EVAL(Q, F.($), $(1)**5*$(2)*$(3)) ;

T=EVAL(R, G.($), SIN(2*$(1)))) ;

今Q, Rとして前の例の結果 $Q=F^{(1^2\,3)}.(A, B, C)$, $R=G^{(1^3)}.(A)$を使用するとすると

$S \leftarrow 20A^3B$, $T \leftarrow -8\cos(2A)$ となります。

b. DIFF Pseudo variable

Format: DIFF(f)=CHAIN(e_1, e_2, ……, e_n)

Result :

n個のvariableからなるfunction variable fの1階偏導関数をexpression e_1, e_2, ……, e_nにassignする。然しながらDIFFはPsedo variable (見かけ上の変数)であるからe_1, e_2, ……, e_nを取り出すことは出来ません。又expression e_1, e_2, ……, e_nはfunction variableのi番目のargumentを$($i$)として使用することが出来ます。もし, 任意の偏導関数を定義する必要のないときはe_iに$を指定します。

<例>

LET(Y=F.(3*X) ; DIFF(F)=CHAIN($(1)**2+A**2) ; R=DERIV(Y, X)) ;

$R \leftarrow 3(9X^2+A^2)$

DIFF(F) より(1)\equiv 3X$となり偏導関数$e_1 \equiv 9X^2+A^2$を作り$R \equiv 3F^{(1)}.(3X)$の$F^{(1)}.(3X)$に$e_1$を代入するとRの結果となります。

<例>

LET(Y=A**2+G.(A+B, 2*A*B) ; DIFF(G)=CHAIN($(1)+$(2), 1-$(2)) ;

Z=DERIV(Y, A)) ;

$Z \leftarrow 3A+B+2AB+2B(1-2AB)$

DIFF(G) よりfunction variable G.(A+B, 2*A*B) の偏導関数e_1=$(1)+$(2)$\equiv A+B+2AB$, $e_2 \equiv 1-$(2)$\equiv 1-2AB$を作り$Z \equiv 2A+G^{(1)}.(A+B, 2AB)+2BG^{(2)}.(A+B, 2AB)$の$G^{(1)}.(A+B, 2AB)$, $G^{(2)}.(A+B, 2AB)$にそれぞれe_1, e_2を代入します。

<例>

LET(Y=F.(B−2*A, B+2*A); DIFF(F)=CHAIN($, SIN($(1)+$(2))) ;
Z=DERIV(Y, A)) ;

Z←−2F(1).(B−2A, B+2A)+2SIN(2B)

$は偏導関数 e_1 を定義しないことを示しています。$e_2 \equiv$ SIN(2B),

$Z = -2F^{(1)}.(B-2A, B+2A)+2F^{(2)}.(B-2A, B+2A)$ に e_2 のみを代入します。

c. DRV

Format : DRV($f(arg_1, arg_2, \cdots, arg_r)$, $(1), n_1, $(2), n_2, ⋯, $(r), n_r)

Result :

n_i は正負の整数で，$(i) は$arg_i$を示しています。この時次の様な結果となります。

$$f^{(1^{n_1} 2^{n_2} \cdots\cdots r^{n_r})}.(arg_1, arg_2, \cdots\cdots, arg_r)$$

なお n_i が省略されると１と見なします。

<例>

LET(Y=DERIV(G.(X**2), X) ; Z=DRV(G.(X**2), $(1))) ;

$Y \leftarrow 2XG^{(1)}.(X^2)$, $Z \leftarrow G^{(1)}.(X^2)$

これはDERIVとDRVとの相異を示しています。即ちDERIVはatomic variable で微分するのに比して，DRVはargument（この例では$(1)≡$X^2$)で微分することを示しています。

<例>

LET(A=DRV(F.(X+Y, Y, X*Y), $(1), 2, $(3))) ;

$A \leftarrow F^{(1^2 3)}.(X+Y, Y, XY)$

LET(DIFF(G)=CHAIN($, $(1)*COS($(2))) ;
D=DRV(G.(X*Y, X−W), $(2))) ;

D←XYCOS(X−W)

この例ではDIFF(G) により e_1 は偏導関数を求めないけど e_2 の偏導関数は

$e_2 \equiv$ XYCOS(X−W) となり，これらをD≡$G^{(2)}$.(XY, X−W) に代入します。

2.4. Comparision : IDENT

Format : IDENT(e_1 ; e_2)

Result :

2つのFORMAC expression e_1とe_2を比較し数式として等しいときIDENTは１となる。等しくないときは０となる。代数的な比較は行ないません。

<例>

LET(X=A+C−D*E+A*SIN(−A)+A ;
Y=−D*E+C+2*A−SIN(A)*A) ;

IF IDENT(X;Y) THEN……

この例では，AUTSIM transformation(Automatic Simplification)と並べかえ(lexicographical recording)によってX←2A−ASIN(A)＋C−DE，Y←2A−ASIN(A)＋C−DEとなり，XとYは数式的に等しいと見なされ，従ってIFステートメントのTHEN以後が実行されます。

<例>

LET(X=3*(A+B); Y=3*A+3*B); BIT1=IDENT(X;Y);
BIT2=IDENT(EXPAND(X);Y);

PL/I variable BIT1=0，BIT2=1となります。BIT1=IDENT(X;Y)は代数的には等しいけれど，同一数式ではありませんのでBIT1=0となります。

2.5. Expression Analysis Routine:

これらのRoutineはFORMAC expressionを分析する時に使用します。expressionの係数関係の処理をするroutine，a/b形のexpressionを処理するroutine，更にexpressionのoperatorの処理をするroutineがあります。

a. COEFF, HIGHPOW, LOWPOW

Format: (i) COEFF(e_1, e_2)
(ii) HIGHPOW(e_1, e_2)
(iii) LOWPOW(e_1, e_2)

Result:

(i) COEFFはexpression e_1の中にあるsubexpression e_2の係数を取り出します。

<例>

LET(Y=A*X**5+2*A*X+(A−B)*(A+B)**2+X**Z+7;
C1=COEFF(Y, X**3); C2=COEFF(Y, X**Z); C3=COEFF(Y, X);
C4=COEFF(EXPAND(X*Y), X); C5=COEFF(Y, A);
C6=COEFF(Y, (A+B)**2));

Result: C1←A, C2←1, C3←2A, C4←$-B^2A+A^2B+A^3-B^3+7$
C5←$2X+X^3$, C6←A−B

(ii) HIGHPOWはexpression e_1の中からsubexpression e_2に関する最も高いpower(ベキ乗)をとり出して来ます。

(iii) LOWPOWはexpression e_1の中からsubexpression e_2に関する最も低いpowerをとり出します。

<例>

LET(Y=X**5+(A+B)**4*X**2+7*LOG(X+1);
P=HIGHPOW(Y, X); Q=LOWPOW(Y, X); R=HIGHPOW(Y, A+B);
S=LOWPOW(Y, A));

P←5, Q←2, R←4, S←0

b. NUM, DENOM

Format: (i) NUM(*e*)

(ii) DENOM(*e*)

Result:

(i) NUMはexpression *e*が*a*/*b*形である時, numerator *a*を取り出します。

(ii) DENOMはexpression *e*が*a*/*b*である時, denominator *b*を取り出します。

なお, 負のベキ乗をもつexpression等はdenominatorとして取り出されます。更にexpression *e*が*a*/*b*形でないときは, NUMは*e*をDENOMは1の値となります。

<例>

LET(P=A/B+C/D; Q=CODEM(P); R=NUM(P); S=DENOM(P);
T=NUM(Q); U=DENOM(Q));

Result: R←A/B+C/D, S←1, T←AD+BC, U←BD

<例>

LET(X=(A+B)/3; Y=NUM(X); Z=DENOM(X));

Y←1/3(A+B), Z←1この例に見られる様にrational numberのnumerator, denominatorは取り出すことは出来ません。

c. Non-algebraic Manipulation Function. LOP, NARGS, ARG

LOP, NARGS, ARGは任意のexpressionの構造に関する情報をうるためのfunctionである。これらのfunctionの目的は, 任意のexpressionが"Lead operator"で関連づけられたargumentのchainであると考えるところにある。即ち

X^2 はLead operator **によって支配されたChain(X, 2)である。

$3AX^2$ は 〃 * 〃 Chain(3, A, X^2)である。

AX^2+BX+C 〃 + 〃 Chain(AX^2, BX, C)である。

このようにexpressionを考えることにより, LOPはLead operatorを, NARGSはArgumentの数を, ARGは*i*番目のargumentをそれぞれとり出すことにある。

(i) LOP

Format: LOP(var). ここでvarはFORMAC variable nameである。

Result:

LOPはFORMAC variable "var"が持っている"Lead operator"によって次の様なPL/I integer(FIXED, (31, 0))をとり出します。

LOP	Lead operator	LOP	Lead operator
0	ERF	3	LOG2
1	LN	4	SIN
2	LOG10	5	COS

LOP	Lead operator	LOP	Lead operator
6	ATAN	32	negative rational constant
7	ATAN(X,Y)	33	negative long integer
8	SINH	34	negative floating constant
9	COSH	35	negative short Integer
10	ATANH	36	zero
11	CHAIN	37	positive rational constant
12	DERIV	38	positive long integer
15	COMB	39	positive floating constant
16	FAC	40	positive short integer
19	STEP	41	#P
21	FNC	42	#E
24	PLUS(+)	43	#I
25	MINUS(−)	44	VAR(atomic variable)
26	TIMES(*)	45	BVAR(S−variable)
31	EXPON(**)	46	atom(new atomic)
		99	undefined Lead operator

<例>

LET(P=3*X**2+4*X+5；Q=A*G.(A+B,C))；LOPP=LOP(P)；
LOPQ=LOP(Q)；

Result：LOPP=24(Code'+'), LOPQ=26(Code'*')

<例>

LET(X=A/B；Y=A−B)；L1=LOP(X), L2=LOP(Y)；

Result：L1=26(Code'*'), L2=24(Code'+')

内部的にはA/B, A−Bはそれぞれ A*B**(−1), A+(−B) と記憶されています。

(ii) NARGS

Format： NARGS(var)

Result：

NARGSはvarにassignedされたexpressionのLead operatorによって結ばれているargumentの数をとり出して来ます。しかもPL/I integer(FIXED,(31, 1))であります。

<例>

LET(P=3*X**2+4*X+5；Q=A*G.(A+B, C)；NP=NARGS(P)；
NQ=NARGS(Q))；NP=3(3X²と 4X と 5), NQ=2(AとG.(A+B, C))となります。

(iii) ARG

Format : ARG(n, e)

Result :

ARGはexpression eのn番目のargumentをとり出して来ます。nは非負の整数。$n=0$の時はeとなります。

<例>

```
LET(P=3**4**2+4*X+5);
DO I=1 TO NARGS(P);
LET(I="I"; OP(I)=ARG(I, P));
END;
```

Result : $P \leftarrow 4X+3X^2+5$, NARGS=3 従ってOP(1)$\leftarrow 4X$, OP (2) $\leftarrow 3X^2$, OP(3)$\leftarrow 5$となりargumentをとり出して来ます。

2.6. Economization of Storage

長いFORMAC expressionは大きなspaceを必要とするので，使用出来る記憶場所がなくなってしまうことがあります。これを防ぐためには2つの方法があります。

a. SAVE

Format : SAVE(var_1 ; var_2 ; … ; var_n)

Result :

VAR_iにassignされているFORMAC expressionは二次記憶装置に移され，それまでVAR_iによって使用されていた記憶場所は解放される。続いてVar_iを使用すると自動的にcoreの中へとり出して来ます。SAVEされていたFORMAC variableを一度使用すると，二次記憶装置のその場所は解放されるので，更にresave するためには今一度SAVE する必要があります。

<例>

```
(i)   LET(A=X+Y);
(ii)  SAVE(A);
        ⋮
(iii) LET(B=A+C);
(iv)  SAVE(A ; B);
        ⋮
(v)   LET(A=X*Y);
```

結果としては(i)ではA$\leftarrow$X+Yとなり，その結果を(iii)によりsecondary storageにcopyされ，Aに使用されていたspace は解放される。(iii)によりAを使用することにより再びsecondary storageからcore の中に取り出して来て，その結果B$\leftarrow$X+Y+Cとなると共にAのexpressionを記憶しているSecondary storageはscratch されます。従っ

てMによりA，Bにassignされているexpressionがsecondary Storageにstore されます。Mによりa←XYとなりsecondary storage のAのexpressionは scratchされます。

b. ATOMIZE

Format : ATOMIZE(var_1 ; var_2 ; … ; var_n)

Result :

FORMAC variable var_nはatomic variableとなり，var_nにassign されていたexpressionは消えてしまうと同時にそのspaceは解放されます。

2.7. FORMAC option : OPTSET

Format : OPTSET($option_1$; $option_2$; …… ; $option_n$)

Result :

$option_n$によって指定された下記のoptionによりセットされる。一度set されると次のOPTSETによりリセットされるまでその効力は残っています。

$option_n$としては次の様なものがあり，指定しないとunderline のついたものが使用されます。

OPTSET option :

1.	TRANS	NOTRANS
2.	INT	NOINT
3.	EXPND	NOEXPND
4.	EDIT	NOEDIT
5.	PROPER	IMPROPER
6.	LINELENGTH＝×××	〔120〕

Effect :

1. TRANSはPL/I Like-function (SIN, LOG etc) のargumentがnumeric constant であるときは，常にその関数値を計算します。
   NOTRANSはsymbolic entityとして残しておきます。
2. INTはFORMAC integer function FAC, COMB, STEPのargument がnumericであるときその関数を自動的に計算します。
3. EXPNDはexpressionの計算をするとき，multinomial及びdistibutiveの法則を適用します。NOEXPND はこの機能を中止させる働きがあります。
4. EDITは結果をprint outするとき，編集して見易い様にします。
   NOEDITはCore に記憶されている形のままプリントアウトします。
5. PROPERはrational numberを$a+b/c$ ($b<c$) の形に直します。
   IMPROPERはa/bの形のままにしておきます。
6. LINELENGTH＝××× は結果をプリントするときに1行に印刷される文字の数を示し

ています。指定しないと 120 文字となります。

〔注意〕

この他にoptionとしてPRINT, NOPRINTがあります。OPTSET(PRINT) はこのステートメントに続くLETステートメントをすべてPRINT OUTステートメントと同じ働きをもたせます。これはプログラム・デバッグイングに役立ちます。OPTSET(NOPRINT) はPRINTの機能を中止させます。なお，このPRINT, NOPRINTの指定は他のoptionと一諸にOPTSETの中で書いてはいけません。

§3 FORMAC-PL/I Interface

ここではFORMAC variable と PL/I variableとの結合，及びError の生じた時の処理の仕方，FORMAC source deckの構造について記します。

3.1. FORMAC statementの中でPL/I variableの使用

a. Double Quates operator

Double Quates operator (") で囲まれたPL/I variableはLET, PRINT_OUT, SAVEの中のFORMAC expression の中において使用することが出来ます。

<例>

PL/I variable A, B, Cがそれぞれ characterで3, 22/7, A*X**2+DERIV(Y, X) であり

```
LET(Y=SIN(X); Z="A"*A*"C"+"B");
```

が実行されると Z← 3A(AX²+COS(X))+22/7 となります。

<例>

次の例はDOステートメントを使用するときに有効であります。

```
DO I=1 TO N; LET(I="I");
   ⋮
LET(A(I)=B(3*I)+I);
   ⋮
END;
```

この中ではDO I=1 TO N;のI及び"I"はPL/I variableであり，LET(I="I") のIはFORMAC variableであります。

<例>

FORMAC expression を PL/I variableとして取りだすためには LET("PL/I var"=FORMAC expression) と指定します。

```
LET(Y=X**2+3*X; "A"=Y+1);
```

この結果，Aは'X**(2)+3*X+1' となります。なお，この時PL/I variable character string AはFORMAC variableを表わしうるに充分な長さを持ってい

なければなりません。

b. Argument Passing

internal又はexternal procedure へ argumentとしてFORMAC variable をpassするための一般的方法はCALL statement の argument Listのvariable をsingle quotes(')で囲んでやる必要があります。

<例>

```
P:PROC(A, B);
  DCL(A, B) CHAR(8);
  LET("B"="A"+3);
  END P;
    :
  CALL P('X+2','BETA');
```

この結果FORMAC variable BETA←X+5

なお，"called" Procedureの方ではFORMAC variableに対応するparameter を PL/I character string variable と Declareし，LET, PRINT_OUT, SAVE statement で使用するときには，常にdouble quote('')で囲んで使用します。

c. CHAREX

Format: CHAREX(var=variable)

Result: PL/I varying character string variable varに右辺の FORMAC variable の character string valueをassignする。

<例>

```
LET(Y=A/B; Z=SIN(3*Y*B)/B); CHAREX(X=Z);
```

X←'Z=B**(-1)*SIN(3*A)'となります。

3.2. PL/Iステートメントの中でのFORMACの使用。

次の2つのステートメントはFORMAC constant を PL/I constantに変換します。

Format: INTEGER(var)
ARITH(var)

Result: FORMAC variable var が numeric constantの時，INTEGERは varの値をfixed point binary(31, 0)number に変換します。varが floating point又はrational numberの時はroudされます。ARITH はvarの値をdouble precision の floating point number にします。

<例>

```
LET(A=4; B=5.2; C=11/4);
X=INTEGER(A)+INTEGER(B)+INTEGER(C)
```

Y＝ARITH(A)＋ARITH(B)＋ARITH(C)

Result：X←4＋5＋3＝12，Y←1.195E01

3.3. FORMAC ON CONDITION

FORMACステートメントを実行中におきるErrorは5つに大別されます。

a. DENSIZE ……使用出来るcore spaceがなくなった場合です。SAVE又はATOMIZEを使用してspaceを解放する。

b. DENSYN……使用したFORMACステートメントの構造上の間違いの時です。

c. DENSEM……使用したFORMACステートメントの機能上の間違いの時です。

d. DENSYSM ……FORMAC System Errorです。

e. DENERRR ……上記のエラーが発生した時にそのメッセージをプリントするためのものです。

上記のエラーが発生した時，特別な処理をするためには次の様にします。

ON CONDITION(DENSYN) GO TO L2;

L2：特別な処理をするステートメント

⋮

なお，ON CONDITIONがなければ上記DENSIZE，DENSYN，DENSEM，DENSYSMのエラーが起きたすぐ後にDENERRRが発生してエラーに関するメッセージをプリントします。

3.4. FORMACプログラムの構成

PL／I FORMACプログラムは，2つのmoduleで構成されている。第1のmoduleはPreprocessorと呼ばれ，FORMAC Languageを，その機能に応じた様々なsubprogramを含んだPL／I Languageにほん訳すると同時にエラー・メッセージをプリントします。

第2のmoduleはFORMAC Libraryと呼ばれ，様々なsubprogramを含んでおり，これらはLink Editorによって実行可能なmoduleの中に組みこまれます。

FORMAC JOB の流れ

FORMAC JOB の流れをステップに分けて考えると次の様になります。